22/09/2023
friday

* **CHAR ARRAYS →**

- **Creation →** char array_name[size];

  char ch[10];

- **Access →** array_name[index];

  ch[index];

- **Input →** cin >> ch;

In char arrays, we need not to use the loops for taking input the characters one by one.

NULL CHARACTER → It represents the termination of the string.
Null character is represented as '\0'.
Its ASCII value is 0.

- **Output →** cout << ch;

As like input, we can also output just by using the cout statement.

NOTE- By default, char arrays is also passed by reference.

__/__/____

Code-

```
#include<bits/stdc++.h>
using namespace std;

int main(){
    char ch[10];
    cin>>ch;
    cout<<"the one and only: "<<ch<<endl;
    // checking ASCII value of null character
    cout<<"ascii value is: "<<(int)ch[6];
}
```

Input - Bharat

Output - the one and only: Bharat
ascii value is: 0

* Delimiter -> It is a special character which indicates the beginning or end of a statement or string.

Delimiter of cin in char arrays and strings ->

- new line character -> '\n'
- tab -> '\t'
- space -> '_'

__/__/____

* **cin.getline( ) →**

When we want to use space and tab while entering the data, we use getline function to take input.

```
cin.getline (array_name, size);
cin.getline (ch, 100);
```

Delimiter of cin.getline( ) →

new line character → '\n'

* **length of string →**

```
#include<bits/stdc++.h>
using namespace std;
int findlength (char ch[], int size){
    int length = 0;
    while (ch[length]!= '\0') {
        length ++;
    }
    return length;
}
int main() {
    char ch[100];
    cin.getline (ch, 100);
    int len = findlength (ch, 100);
    cout<<"length of string is:"<<len<<endl;
}
```

Input → nisha
Output → length of string is: 5

- strlen( ) is the inbuilt function used for finding string length.

__/__/__

* Reverse a string →

```
#include <bits/stdc++.h>
using namespace std;

void reverseString (char ch[]) {
        int i= 0;
        int j = strlen(ch) - 1;

        while (i<=j) {
            swap (ch[i], ch[j]);
            i++;
            j--;
        }
    }
int main( ) {
    char ch[100];
    cin.getline (ch, 100);

    reverseString (ch);
    cout <<"reverse printing : "<< ch <<endl;
}
```

Input → nisha kashyap

Output → payhsak ahsin

Time complexity = O(n).

- strrev( ) is the in-built function used for reverse a string.

__/__/__

* Upper case conversion →

```
#include <bits/stdc++.h>
using namespace std;

void convertToUpperCase (char ch[]){
    int index= 0;
    while (ch[index] != '\0') {
    //condition whether the character is in lower case
    // then convert to upper case
    if (ch[index] >='a' && ch[index] <='z'){
        ch[index]= ch[index] -'a' +'A';
        }
    index ++;
    }
}
int main( ){
    char ch [100];
    cin. getline (ch, 100);

    convertToUpperCase (ch);
    cout << "upper case: " << ch;
}
```

Input → Char Arrays And Strings

Output → CHAR ARRAYS AND STRINGS

Time complexity → O(n)

__/__/____

* Replace character →

```
#include<bits/stdc++.h>
using namespace std;

void replacecharacter(char ch[]){
    int index = 0;
    while(ch[index] != 0){
        // replace the character @ with space' '
        if(ch[index] == '@'){
            ch[index] = ' ';
        }
        index++;
    }
}
int main(){
    char ch[100];
    cin.getline(ch, 100);
    replacecharacter(ch);
    cout << " replaced: " << ch;

}
```

Input → My@Name@is@Nisha

Output → My Name Is Nisha

Time complexity → O(n)

__/__/___

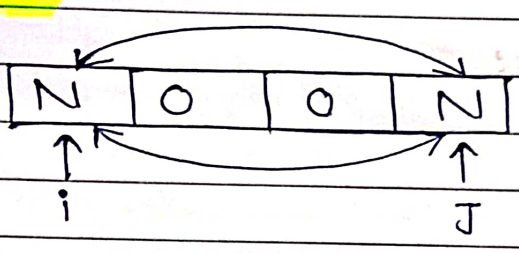
* **Check Palindrome →**

Palindrome → If string remains the same if we write it either left to right or right to left, then it is palindrome.

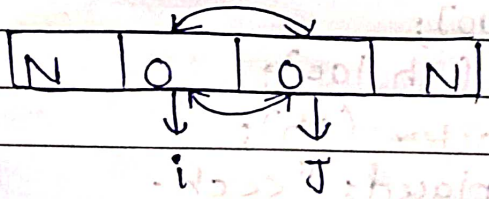
Input → NOON

Output → Valid palindrome

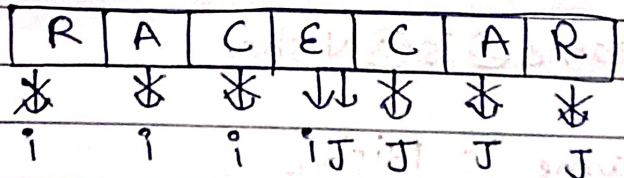
logic → NOON




```
if (ch[i] == ch[J]){
    i++;
    J--;
}
```



RACECAR




```
while (i<=J) {
    if (ch[i] == ch[J]){
        i++; J--;
    }
    else
        return false;
}
return true;
```

__/__/__

Code →

```
#include<bits/stdc++.h>
using namespace std;

bool checkPalindrome(char ch[]){
    int i=0;
    int j=strlen(ch)-1;

    while(i<=j){
        if(ch[i]==ch[j]){
            i++;
            j--;
        }
        else{
            return false;
        }
    }
    return true;
}
int main(){
    char ch[100];
    cin.getline(ch,100);
    bool isPalindrome = checkPalindrome(ch);

    if(isPalindrome){
        cout<<"valid palindrome"<<endl;
    }
    else{
        cout<<"invalid Palindrome"<<endl;
    }
}
```

Time complexity - O(n)

__/__/____

* **STRINGS →**

Sequence of characters. Unlike array, whose size is static, the size of strings can be changed at runtime i.e. they are dynamic in nature.

- **Creation →** string name;
- **Access →** name[index];
- **Input →**

= cin >> name;
Taking input like this will not read space ' ', new line character '\n' and tab '\t'.

= getline (cin, name);
This will read space ' ' and tab '\t'.

- **Output →**

Cout << name;

__/__/____

Code →

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    string name;
    getline (cin, name);
    cout << "Printing string: " << name << endl;
    cout << "printing first character: " << name[0] << endl;
    cout << "printing first character ASCII value:" <<
        (int) name[0] << endl;

    cout << "using loop for printing characters: " << endl;
    int index = 0;
    while (name[index] != 0) {
    cout << "index:" << index << "character: " << name[index]
                                                << endl;

        index ++;
    }
    cout << "print 5th character: " << name[5] << endl;
    cout << "printing 5th character ASCII value:" <<
                                    (int) name[5] << endl;
}
```

Input → nisha

Output →

```
printing string : nisha
printing first character : n
printing first character ASCII value : 110
using loop for printing characters :
index : 0   character : n
index : 1   character : i
index : 2   character : s
index : 3   character : h
index : 4   character : a

Printing 5th character : '_'
Printing 5th character
ASCII value : 0
```

__/__/____

* In-Built functions in strings ⇒

(1) name.length() = To find string length.

(2) name.empty() = To check whether string is empty or not.

(3) name.at(index) = To access particular index of string.

(4) name.front() = To find front character.

(5) name.back() = To find back character.

Code →
```
#include<bits/stdc++.h>
using namespace std;

int main(){
    string name;
    getline(cin,name);
    cout<<"string length: "<<name.length()<<endl;
    cout<<"empty or not: "<<name.empty()<<endl;
    cout<<"accessing character: "<<name.at(4)<<endl;
    cout<<"front character: "<<name.front()<<endl;
    cout<<"back character: "<<name.back()<<endl;
}
```

I/P = nisha

O/P =
```
string length : 5
Empty or not: 0
accessing character: a
front character: n
back character : a
```

__/__/____

- **Append string →** To concatenate two strings.

str1.append(str2);

Code →

```
#include<bits/stdc++.h>
using namespace std;
int main(){
string str1="love";
string str2="Babbar";
cout<<"Before append"<<endl;
cout<<str1<<endl;
cout<<str2<<endl;

cout<<"after append"<<endl;
str1.append(str2);
cout<<str1<<endl;
cout<<str2<<endl;

}
```

Output →

```
Before append

love
Babbar

After Append

loveBabbar
Babbar
```

__/__/____

- **Erase String characters →**

Code →
```
#include <bits/stdc++.h>
using namespace std;

int main() {
    string str = "This is an example sentence";
    str.erase (10,8);
    cout << str << endl;

    str.erase (str.begin()+9);
    cout << str << endl;

    str.erase (str.begin()+5 , str.end()-8);
    cout << str << endl;
}
```

Output →
```
This is an sentence.
This is a sentence.
This sentence.
```

- **Insert →**

Code →
```
#include <bits/stdc++.h>
using namespace std;
int main() {
    string str1 = "Touch sky with glory";
    string str2 = "the ";
    str1.insert ( 6, str2);
    cout << str1 << endl;
}
```

Output → Touch the sky with glory.

__/__/__

- **Push and Pop back →**

Code →
```
#include<bits/stdc++.h>
using namespace std;
int main(){
    string str1="love";
    str1.push_back('r');
    cout<<str1<<endl;
    str1.pop_back();
    cout<<str1<<endl;
}
```

Output →
```
lover
love
```

- **Find →**

Code →
```
#include<bits/stdc++.h>
using namespace std;
int main(){
string str1="Touch the sky with glory";
string str2="sky";

if(str1.find(str2)==string::npos){
    cout<<"not found"<<endl;
    }
else{
    cout<<"found"<<endl;
    }
}
```

Output → found.

Note → npos returns true if there are no matches present else it returns false.

__/__/____

- **Compare →**

Code →

```
#include <bits/stdc++.h>
using namespace std;
int main(){
  string str1 = "nisha";
  string str2 = "nisha";
  if (str1.compare(str2)==0){
      cout<<"matching"<<endl;
  else{
      cout<<"not matching"<<endl;
    }
}
```

Output → matching

- **Sub String →**

Code →

```
#include <bits/stdc++.h>
using namespace std;

int main(){
  string str = "Touch the sky with glory";
  cout<< str.substr(10,3)<<endl;

}
```

Output → sky